Apache ZooKeeper Poster

In short, ZooKeeper was created in the effort to maintain and develop an open-source server that lets you coordinate highly reliable distribution.

Apache ZooKeeper™

What is Zookeeper

Zookeeper is centralized around maintaining and configuring information, synchronizing distributed data, naming, and providing services in groups. These different services are used in one way or another by distributed applications. Each time a different service is used a lot of work goes into race conditions, bug fixing, and software problems that are inevitable.

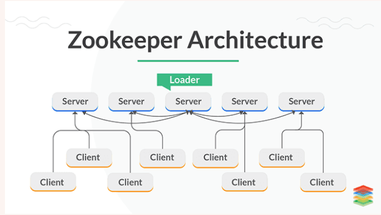
With the difficulty of implementing the different services, initially, applications turn out to scrimp on the services. This makes applications brittle when any change is made and leads to applications being difficult to manage. Even if the changes are done correctly, implementations of the services can lead to management complexity when the application is deployed.

Created by Enrico Dreyer 31210783

Typical functionality

- When the ZooKeerper servers start, it waits for all clients to connect to that server.
- After they have connected, the clients will connect to the nodes and the node will either be a leader or a follower.
- Once the client is connected, the node will get a session id, and the node will send an acknowledgment to the client that it is connecting to.
- If there is no response from a server, the node will resend a message to another server.
- If there is a response, a consistent acknowledgment will be sent to make sure the connection is not lost.
- The client can then perform functions as needed.

Zookeeper Architecture



The architecture is developed to work on client-server architecture. Clients are machine nodes and servers are nodes

In the diagram, you can see that every client sources a client library.

You can also see that each client source can also communicate with anyone of the Zookeeper nodes

ZooKeeper Use Cases

- Managing the configuration.
- Naming the services.
- Choosing a leader
- Queuing messages
- Managing notification systems
- Synchronization

A way that ZooKeeper uses to communicate is by using a CLI. This gives a user the feature of using various options as well as the ability to debug.

Kafka Support

ZooKeeper was developed for distributed systems to synchronize their services as well as a naming registry. When Zookeeper is working with Apache Kafka, it is primarily used to follow up on the statuses of nodes in a cluster of Kafka and maintain a listing of Kafka messages and topics.

As of June 2021, Kafka services can not be used without the installation of Zookeeper, even if the use case requires one broker, or one partition, or one topic.

Any distributed system needs a way to coordinate its tasks. Zookeeper was used to satisfy the need for Kafka to coordinate its tasks. When looking at MongoDB or Elasticsearch, they have their own mechanisms to coordinate their tasks.

Apache Zookeeper Features

1. Naming service
2. Updating the nod's statuses
3. Managing clusters
4. Automatic failure recovery
5. Simplicity
6. Reliability
7. Ordered
8. Speed
9. Scalability
10. Higher-level abstraction
11. It is fast
12. Ordered messages
13. Serialization
14. Reliability
15. Atomicity
16. Sequential Consistency
17. Single System Image
18. Timeliness

Zookeeper Architecture Components

Client: This is used to access information from its server. The client sends messages to the server in order to let the server know that the client is alive, if there is no response from the server it connects to another server automatically until it gets a response.

Server: The server works in a way that gives acknowledgment to the client to let the client know that the server is alive, the server also provides the services that the client needs.

Leader: When a server node fails, the server node recovers that node automatically.

Follower: The server node that follows the commands that the leader gives.

Newer or Similar applications tools than ZooKeeper

Kubernetes: This is an open-source orchestration system for docker container

Consul by HashiCorp: This is a multi-datacenter, distributed, and highly available service discovery system with health checking and key/value storage.

Cycle.io: This is a SaaS platform designed to make running containerized applications in the cloud simple.

etcd: This is a strong and consistent, reliable way to store data that needs to have access by a distrabuted system.